The last decade has witnessed an extraordinary evolution in the way that computing devices have affected our lives. They are far more pervasive and context driven than we could have ever imagined. The World Wide Web has been transformed from being a humble collection of linked pages to one displaying dynamic behavior based on context and user actions. Web sites today are capable of displaying responsive design, changing layouts based on the resolution of the target device, delivering content based on the location of the user, showcasing media without having the user download and install any media-specific plug-in, and the list goes on. The next decade will be the one where an immersive and fluidic experience for the client will drive innovation.

ASP.NET was a big leap forward in the way dynamic web sites could be rapidly built using Visual Studio as the tool of choice for rapid application development. We are pretty sure that you still like the way you can drag and drop ready-to-go controls on the web application design surface and bind them to data sources in just a few clicks. The evolution of ASP.NET since its inception has also been phenomenal, and to keep pace ADO.NET has seen quite a few changes as well. Today it supports a multitude of data access paradigms including WCF (Windows Communication Foundation) Data Services for REST (Representational State Transfer)-style Create, Read, Update, and Delete CRUD operations.

In this chapter, you will get a glimpse at the modern data access paradigms that will help you learn the ways in which the data access components are designed and, in turn, use them to suit your application requirements. This will be at a very high level without getting into any last-mile implementation details. The concepts discussed here will be used throughout the rest of the book to help you gain an understanding of how the different components of the paradigm come together to build data-driven web sites.

Note Data access paradigms are discussed at length in Chapter 8.

Given the overview, here is what we are going to cover:

- What are the new data access paradigms?
- What are the different data sources that you can use with ASP.NET 4.5 web sites?
- What are the enhancements to ADO.NET?
- How to create your first data-driven page in ASP.NET 4.5 and ASP.NET MVC 4.

If you are unsure if data-driven web sites are a good idea, then we would strongly recommend reading the book *Beginning ASP.NET 2.0 Databases: From Novice to Professional* by Damien Foggan (Apress, 2006). It is a nice precursor to the material discussed in this book, and you will easily be able to relate to the newer concepts discussed here.
The New Data Access Paradigms

How do you decide the approach you want to take to building a data access layer in your development workflow? Fortunately, the decision is fairly simple at a very high level in the design process, and it is driven by only a couple of possible scenarios:

**The database already exists:** In this case you can choose to generate an Entity Model from the existing database and then use the Entity Model to create the data access layer.

---

**Note** *Entity Model* is an Entity Relationship Diagram generated from the database schema typically with the help of a conceptual model generator. Microsoft .NET Framework 4.5 features a built-in model generator in the ADO.NET Entity Framework.

---

**The database is newly created:** In this case, there are again a couple of options. You could start by creating the Entity Model first and then use it to drive the steps to create the database. You could also create the database first and then autogenerate the Entity Model as described before.

---

**Tip** It is a good idea to start with the Entity Model–First approach. This way you have an opportunity to play with the conceptual model until it is frozen without having to change the database schema every time and avoid annoying the DBA in your team.

The preceding scenarios are driven by a design-centric approach to application development. You also have the choice of a code-centric approach, and a lot of developers prefer that since it is easy to manipulate in code than to modify the model on the design surface. Another argument in favor of the code-centric approach is the ease of deployment; however, you need to be careful since you may end up writing a lot of code!

For a new database, the code-centric approach is simpler than it is for an existing database. There are tools to generate code for an existing database; however, it is error prone and you could lose the flexibility of having the code structured in the fashion you want.

---

**Note** The code-centric approach is gaining popularity in developer communities as it provides a greater degree of flexibility and control. Developers feel empowered, and it is often easier for multitargeted deployment in scenarios where the database (or a subset of it) could be a part of an isolated storage in a handheld device in addition to the database server.

---

To summarize, the three paradigms of data access in modern-day application development are illustrated in the following:

**Database First:** Reverse engineer the Entity Model from the database schema. Generate the entity classes automatically from the Entity Model using tools. Figure 1-1 demonstrates this data access paradigm.